**WEST UNIVERSITY OF TIMIŞOARA**
**FACULTY OF MATHEMATICS AND COMPUTER SCIENCE**
**MASTER STUDY PROGRAM: Big Data - Data Science, Analytics and Technologies**

# Apriori Algorithm for Recommender Systems

**Coordinator:**
Roxana Mafteiu-Scai
**Team:**
Team 1

**TIMIŞOARA**
**2023**

# Team members


- Adrian Tuns
- Iulia Hurloi
- Paul Pop
- Vittoria Piatti


# Theme

Implementing the Apriori algorithm and apply it in a movie recommender system.

# Contents

# Chapter 1

# Introduction

## 1.1 Problem Description

Recommender systems are used in the vast majority of domains, they are implemented using machine learning in order to automate the process of analysing a set of data and extracting an item based on certain criteria.

Specifically in the area of movie recomandation systems, the importance of associations between a set of movies is the base of a good prediction. Personalized movie recommendations are becoming more and more in demand as online movie streaming services expand. Finding movies that suit their interests can be difficult due to the abundance of movies available. Traditional techniques of movie suggestion, such those based on popularity or genre, have a limited capacity to deliver reliable recommendations. Consequently, a more advanced recommendation system is required in order to offer recommendations.

The overwhelming number of movies available on online streaming platforms can make it difficult for users to find movies that match their interests and preferences. Traditional movie recommendation systems, such as popularity-based or genre-based recommendations, are limited in their ability to provide accurate recommendations as they do not take into account individual user preferences. This results in users potentially wasting their time and money on movies that do not align with their interests. There is a need for a more sophisticated recommendation system that can provide personalized recommendations to users based on their movie preferences and ratings, in order to improve the user experience and satisfaction with movie streaming platforms. The solution is to obtain a good recommender system based on the Apriori algorithm.

## 1.2 Project Motivation

Streaming services have grown in popularity, and consequently, movie recommendation systems have become increasingly popular.

Apriori is an algorithm used for item frequency mining with applications in many domains where finding the trend is in question. When it comes to massive amount of data, this algorithm has become the most effective association rule algorithm to extract valuable information, knowledge, and patterns. [**2**]

This project's built movie recommendation system will offer customers a distinc-

tive movie recommendation experience by combining personalized data analysis and data mining techniques. In the case of recommendation system for movies Apriori's algorithm capability of finding association rules is crucial.

# Chapter 2

# Related Work

## 2.1 Related Work

In the research paper "Movie Recommender System Using K-Means Clustering AND K-Nearest neighbor"[**1**] the authors propose a combined approach on the problem. The system receives the user information (preferences, ratings for previously watched movies), after processing the raw data K-mean clustering algorithm, with the module Within Clustered Sum of Square, is applied to separate the movies in genre. The correlation between users and the average rating given by them to every cluster is then calculated. Finally, the prediction is made by applying K-Nearest Neighbor.

Businesses from different sectors seek associative rules or relationships between the items they sell during the process of Knowledge Discovery in Database (KDD)[**3**]. The Apriori Algorithm, helps in forming possible combination item candidates, then tests whether the combination meets the minimum support parameters and minimum confidence which is the threshold value given by the user.

In another paper[**4**] the recommendation system is based on the MovieLens dataset (title and 19 genres for movies dataset and rating and timestamp for ratings dataset). The personalized recommendation is possible after creating users' profiles based on their feedback on different movies. After users' reviews are splitted in liking (more than or equal to 2.5) and disliking (less then 2.5) categories, the apriori algorithm is applied is order to find preference lists for each person.

## 2.2 Conclusions

Analysing the bibliography the conclusion is that applying machine learning in data analysis automates the process of recommendation in any domain. The related work in the field of movie recommendation systems has demonstrated the importance of using sophisticated algorithms to generate personalized recommendations for users.

The results of this project confirmed the findings of the related work, with the Apriori algorithm demonstrating its ability to generate high-quality, personalized recommendations for users based on their preferences and ratings. A similar approach as in the above cited research paper[**4**] will be used, applying on the data set the implementation of the Apriori algorithm.

# Chapter 3

# Apriori Algorithm

There are numerous algorithms for mining association rules. The primary method used in data mining to identify common itemsets is association rules. Association rules are "if-then rules" with two metrics that express the rule's support and confidence for a particular collection of facts. As stated in (Hegland, 2007), Apriori is the first and possibly most significant algorithm for effective association rule discovery. It is employed to obtain the association rule for knowledge discovery and to retrieve frequent itemsets from enormous databases (Al-Maolegi and Arkok, 2014). Due to the excessive number of database scans, ( Singh, Ram, Sodhi, 2013) mention that this "traditional technique" is inefficient. Additionally, if the database is vast, scanning it takes too long. Even so, this algorithm fits the problem we want to solve, more precisely, to create a Movie Recommender Systems.

The Apriori algorithm in our code starts by finding all length-1 frequent item sets (i.e. items that occur more frequently than a minimum support threshold). This is done by iterating through the dataset and counting the support of each unique entry. The support of an item set is defined as the number of transactions in the database in which all items in the set appear. Only item sets that have support greater than or equal to the minimum support threshold are considered as frequent item sets.

After finding the length-1 frequent item sets, the algorithm generates item sets with length 2 by combining length-1 frequent item sets. This process is repeated to generate item sets with length 3, 4, and so on, until no more frequent item sets can be found. To efficiently perform this process, the Apriori algorithm uses the "Apriori property", which states that all subsets of a frequent item set must also be frequent. This property allows the algorithm to prune the search space by eliminating infrequent item sets, thus reducing the computational cost of the algorithm. The result of the Apriori algorithm is a set of frequent item sets, along with their absolute support counts.

In conclusion, the Apriori algorithm is a useful tool for finding frequent item sets and association rules in transactional databases. The algorithm has been widely used in various applications, and its implementation is straightforward. The implementation shown in the code provides a basic example of how the Apriori algorithm can be applied to a database.

## 3.1 Data set

The data set was provided in the form of a text file, containing 8896 lines that represent each an user list of favorite movies.

Figure depicting some entries of the data set:

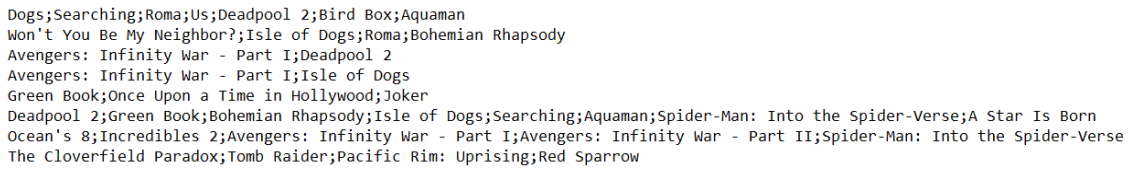


Figure 3.1: Data Set Sample

# Chapter 4

# Solution

## 4.1 Technologies

In order to obtain the recommender system, Python programming language will be used.

Python was chosen both for its simplicity and for its usability in scientific computing.

## 4.2 Setup Instructions

The project was implemented completely using Python 3.x version, using the following libraries: "csv", "operator", "itertools" and "collections".

In order to install and run the application, a system capable of running Python 3.x is needed, requiring therefore:

- Operating System: Linux- Ubuntu 16.04 to 17.10 or Windows 7 to 10
- RAM Memory: 2GB (4GB preferable)

The recommended way of running the project on a system where Python 3.x version is installed is the following:

1. Open the project in an IDE (e.g. PyCharm IDE)
2. Configure a new interpreter using the locally installed Python distribution
    - Recommendation: create a virtual environment on which the corresponding libraries will be installed
3. Install the required libraries ("csv", "operator", "itertools" and "collections"), using pip or other package installers
4. For using the application from a terminal, without the GUI part:
    - Go at the end of the file, and uncomment one of the desired function calls, from the three main functions:
        - start_data_mining() - used to start the actual data mining

  - get_length_itemsets(1) - used to get the n-length itemsets; Note: replace "1" with desired "n" value
  - movie_recommendation() - used to make a movie recommendation

5. For using the application with the GUI part:

   - Go to the gui.py file and run it

## 4.3 Implementation

Our code aims to create a movie recommendation system, based on the results of the Apriori algorithm. Although this project can be carried out in countless ways, we followed in as much detail as possible the steps and sub-objectives predefined by the didactic staff. Therefore, we aimed to reach the following: a. Output all the length-1 frequent movies with their absolute supports, b. Write all the frequent itemsets along with their absolute supports, c. Give a movie recommendation based on the result of the Apriori algorithm (based on predefined input or user input).

In the following, we will explain the design and functionality of the code. In the main, the code is organized as follows: a function for each individual sub-objective, so it is easy to follow by people outside the project; comments that explain certain sections of the code; consistent formatting throughout the code, such as consistent line spacing and indentation, in order to make the code more readable and easier to follow the logic.

This project is based on a pre-established database, so the data-mining process from platforms specific to movie recommendations was not necessary. We started this project with the help of a notebook, where it was necessary to manipulate the dataset in order to apply the Apriori algorithm. However, this method was left behind and the project continued in PyCharm. The results after applying the algorithm were identical for both methods, so we are sure that they worked in accordance. To fulfill the sub-objectives and implicitly, the main objective, our code is based on the following 3 functions: a. `start_data_mining`, b. `get_length_itemsets`, c. `movie_recommendation`. These 3 functions are the basis of our project and help us achieve the main and final objective: creating a recommendation system using the Apriori algorithm.

Those previously mentioned provide the following functionalities:

(a) The `start_data_mining` function is responsible for performing data mining on the provided dataset. First, it opens the movies.txt database, drops the first row (this was previously added for the notebook code to work) and creates a list that will contain the recommendation of each individual user. Besides all this, this function implements the Apriori algorithm, which starts by counting the support of each item in the transactions and only keeping the items that have a support greater than or equal to the minimum relative support. The minimum support set is a requirement, and the minimum confidence level was decided in order to have a result as close to reality as possible. The Apriori Algorithm uses a minimum relative support as input to determine the minimum number of transactions that an item set must appear in to be considered frequent. These items are then combined into sets of length 2, and

their support is counted. The process repeats until no new frequent sets can be found. The resulting frequent sets are stored in a file named 'patterns.txt'. The function returns the set of frequent itemsets.

```
list_of_csv = list(csv_reader)
apriori_result = apriori(dataset=list_of_csv, min_relative_support=0.05)
compute_association_rules(dataset=list_of_csv, apriori_result=
    apriori_result, min_confidence=0.6)
```

Listing 4.1: `start_data_mining`

(b) The `get_length_itemsets` function reads the 'patterns.txt' file created and retrieves all the frequent itemsets of a specified length. Besides this, it takes in a parameter `itemset_length` which specifies the desired length of frequent itemsets to be retrieved. The function loops through the list of frequent itemsets, checking the length of each set, and adds the sets to a new list if the length of the set is equal to the desired length specified in the `itemset_length` parameter. Finally, the function returns the new list containing the frequent itemsets of the desired length.

```
with open('patterns.txt', 'r') as f:
    for line in f:
        if line.count(';') == itemset_length:
            length_itemset_found = True
            print('[' + line.split(';', maxsplit=1)[1][0:-1] + ']')
```

Listing 4.2: `get_length_itemsets`

(c) The function works by looping through the set of association rules and checking if the antecedent (the "left-hand side") of each rule is a subset of the items the user has watched. If this is true, the function adds the consequent (the "right-hand side") of the rule to a list of recommended items. The recommended items list is then returned as the output of the function. This recommendation system is based on the idea that if a user has watched a certain set of movies, they are likely to watch another movie that is frequently watched by people who have watched the same set of movies. The function takes two inputs, the user's favorite movies and the minimum confidence, and returns a list of recommended movies. The recommended movies are the consequence of the association rules.

```
association_rules = []
with open('associationRules.txt', 'r') as f:
    for line in f:
        rule = line.strip().split(':', maxsplit=1)[1].split('->')
        association_rules.append([[i.lower() for i in rule[0].split(';')],
            rule[1].split(';')])

unique_movies = []
with open('oneItems.txt', 'r') as f:
    for line in f:
        movie = line.strip().split(':', maxsplit=1)[1]
        unique_movies.append(movie)

print('\nWelcome to movie recommendation!\nHere is the list of movies from
    our database: ' +
```

```
14          str(unique_movies)[1:-1] + '\nIf you want to exit the program,
                input "quit".\n')
```

Listing 4.3: `movie_recommendation`

## 4.4 GUI

A graphic user interface was generated in order to exemplify the way such an application would look like and function.

There are many options for GUI implementation for Python programs. Our work focuses on simplicity and functionality and the design to support the features of our project didn't require a high degree of complexity.

Tkinter allows to overcome the extreme simplicity of PySimpleGUI and gives access to many resources which allow a more design-focused style of coding.

Tkinter widgets are the building blocks of this system, and we used them to present an interface that follows the requirement of: a. Output all the length-1 frequent movies with their absolute supports, b. Write all the frequent itemsets along with their absolute supports, c. Give a movie recommendation based on the result of the Apriori algorithm (based on predefined input or user input).

It starts with a screen in which there is a button to launch the program and start the mining for the construction of the apriori rules. The position of the button is near the title, it's text states the purpose of it.

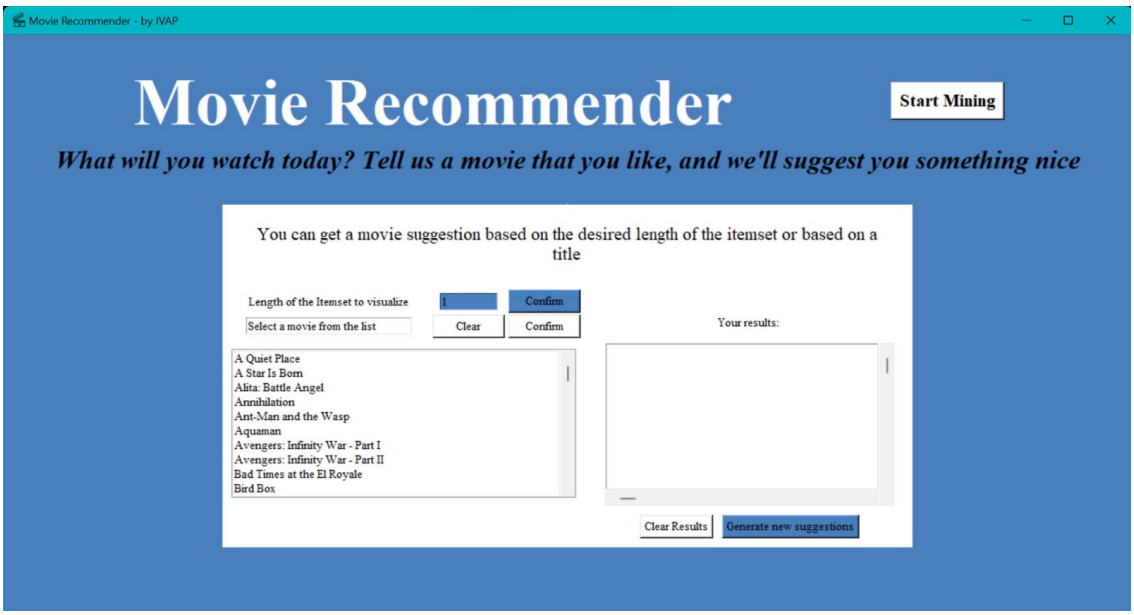


Figure 4.1: GUI Main View

After the 'Start Mining' button is clicked the user can input the length of the item set they want to see as a result of the recommendation system which will be based on the list of movies the user selects from the list.

To validate and start the algortihm the user needs to press the 'Confirm' buttons situated in the near right of the text boxes. The results will appear in the area on the right side on the screen.

The blue button under the results area is connected to a function that will simply erase all the inputs/outputs of the last calculation to allow a new suggestion to be made based on new inputs.

## 4.5 Evaluation

The application was evaluated in a time based manner, and in a code comparison manner.

The measured execution time of the two important functions are the the following:

- for running the apriori algorithm, together with the process of saving to files: 21.25297236442566 s
- for computing the association rules and saving them into a file: 0.3769972324371338 s

The results of the apriori implementation, used in the recommender system, were compared with the results of the apriori algorithm implemented in the library "mlxtend", from the "frequent_patterns" package. After running both forms of the apriori algorithm on the same data set a conclusion was reached, that the results are similar.

The association rules generated by the implementation from the project were also compared with the results of "association_rules" from the library "mlxtend", from the "frequent_patterns" package. The highest confidence rules were the most similar.

# Chapter 5

# Results

```
Welcome to movie recommendation!
Here is the list of movies from our database: 'A Quiet Place', 'A Star Is Born', 'Alita: Battle Angel', 'Annihilation', 'Ant-Ma
 Bandersnatch', 'Bohemian Rhapsody', 'Captain Marvel', 'Crazy Rich Asians', 'Deadpool 2', 'Eighth Grade', 'Fantastic Beasts and
 'Joker', 'Jurassic World: Fallen Kingdom', 'Love, Simon', 'Mission: Impossible - Fallout', "Ocean's 8", 'Once Upon a Time in H
 Far from Home', 'Spider-Man: Into the Spider-Verse', 'The Ballad of Buster Scruggs', 'The Cloverfield Paradox', 'Tomb Raider',
If you want to exit the program, input "quit".


Input some movies that you enjoy, separating them by a ";":

Incredibles 2 ; Deadpool 2
The movie recommendation result: 'Avengers: Infinity War - Part I'


Should we try giving a new recommendation? Answer with: "yes" or "no"
yes
The movie recommendation result: 'Spider-Man: Into the Spider-Verse'


Should we try giving a new recommendation? Answer with: "yes" or "no"
yes
The movie recommendation result: 'Avengers: Infinity War - Part II'


Should we try giving a new recommendation? Answer with: "yes" or "no"
yes
The movie recommendation result: 'Ant-Man and the Wasp'


Should we try giving a new recommendation? Answer with: "yes" or "no"
yes
The movie recommendation result: 'Captain Marvel'


Should we try giving a new recommendation? Answer with: "yes" or "no"
yes
The movie recommendation result: 'Spider-Man: Far from Home'


Should we try giving a new recommendation? Answer with: "yes" or "no"
yes
We are sorry, but no recommendation could be found for your input.
```

Figure 5.1: Terminal Results

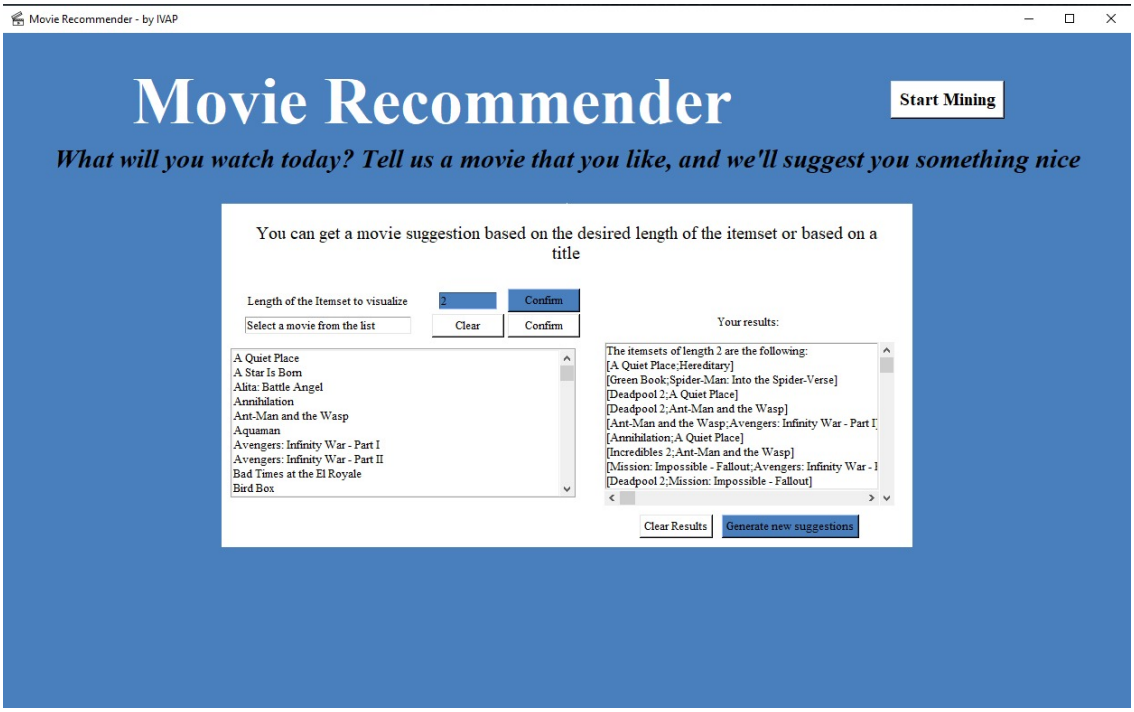


Figure 5.2: ItemSet Output

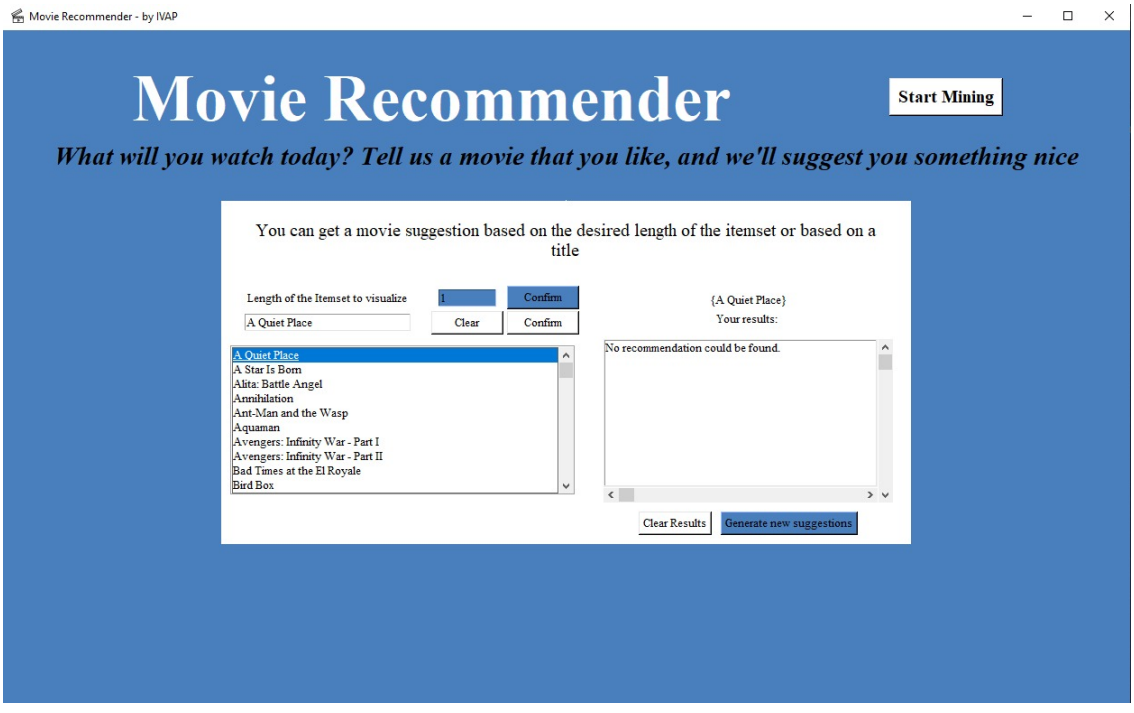


Figure 5.3: One movie Negative Result

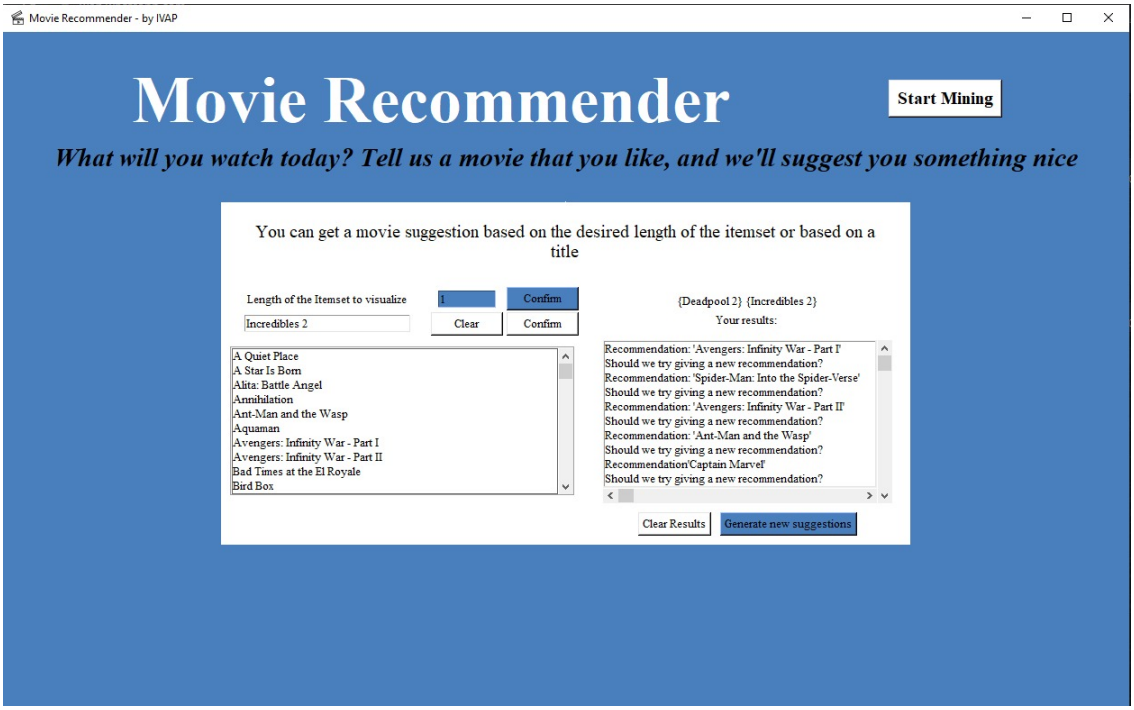


Figure 5.4: Two Movies Input Results

# Chapter 6

# Conclusion and Future Work

The purpose of the project, to implement the Apriori algorithm and to generate an interface in order to visualise the recommender system for movies, was fulfilled.

For future reference the following modifications should be implemented:

- Refactor the source code to obtain better results, taking into consideration the lower length item sets
- Implement more sophisticated and modern looking graphic user interface that is more intuitive and pleasing to the eye of the user
- Improve capabilities of the application by attempting to implement a similar system as the most famous streaming service, a two-tiered row-based ranking system.

# Chapter 7

# Distribution of Work and Efforts

The effort was distributed equally based on our main strengths in order to optimise the division of work and achive satisfactory results.

- Configuration Engineering - Adrian Ioan Tuns, with main focus on back end
- Data Analyst - Paul Pop, with main focus on research, gather information, testing and help in documentation and presentation
- User Interface Expert - Vittoria Piatti, with main focus on front end
- Bussiness Analyst - Iulia Hurloi, with main focus on documentation, presentation and coordination of the activities

Time spent:

1. Individual research on subject: 5h
2. In meeting for analysing and discussing information found will research part: 20h
3. Coding back-end and front-end: 20h
4. Testing: 2h
5. Documentation: 10h
6. Presentations: 3h

# Bibliography

[1] R. Ahuja, A. Solanki and A. Nayyar, "Movie Recommender System Using K-Means Clustering AND K-Nearest Neighbor," 2019 9th International Conference on Cloud Computing, Data Science and Engineering (Confluence), 2019, pp. 263-268, doi: 10.1109/CONFLUENCE.2019.8776969.

[2] Y. Xu, R. Zhan, G. Tan, L. Chen, B. Tian, An improved apriori algorithm research in massive data environment, in: Cyber Security Intelligence and Analytics, Springer International Publishing, 2019, pp. 843–851. by user profiling using apriori algorithm. Applied Soft Computing, 106, 107272. https://doi.org/10.1016/j.asoc.2021.107272

[3] Suprianto Panjaitan et al 2019 J. Phys.: Conf. Ser. 1255 012057

[4] Singh, P. K., Othman, E., Ahmed, R., Mahmood, A., Dhahri, H., and Choudhury, P. (2021). Optimized recommendations